AD A102266

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER final, part 2 | 2. GOVT ACCESSION NO. AD-A102 | 3. RECIPIENT'S CATALOG NUMBER 266 |
| 4. TITLE (and Subtitle) (6) Cost/Performance Comparisons for Programs on Different Machines, (10) | | 5. TYPE OF REPORT & PERIOD COVERED final. 6/1/80-8/31/81 |
| (12) | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Anneliese K. von Mayrhauser Dennis E. Witte | | 8. CONTRACT OR GRANT NUMBER(s) Office of Naval Research (15) N00014-80-C-0364 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Illinois Institute of Technology Chicago, IL 60616 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR064-640 (474) |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research 800, N. Quincy St. Arlington, VA 22217 Code 474 | (11) | 12. REPORT DATE 31 Aug 81 |
| | | 13. NUMBER OF PAGES 21 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research Branch Office 536, S. Clark St. Chicago, IL 60605 (12) 23 | | 15. SECURITY CLASS. (of this report) Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)
approved for public release; distribution unlimited
(9) Final rept. (Part 2), 1 Jun 80-31 Aug 81

LEVEL

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC
SELECTE
JUL 3 1 1981
S
C

18. SUPPLEMENTARY NOTES
To be presented at ECOMA-9, Oct. 6-9, 1981, Copenhagen

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
minicomputers, performance comparison, service selection, utility assessment, finite element method

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Due to a wide selection of machines with vastly different charging algorithms, the question arises "which machine is the most cost-effective". The program's behavior and the impact on charges on different machines need to be explored. Accuracy and correctness of the results and the system's reliability also need consideration. Human factors influence quality as well. A procedure and system is discussed which evaluates these factors and their relationship with each other.

DTIC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

175400

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# Cost/Performance Comparisons for Programs on Different Machines

Anneliese von Mayrhauser
Dennis E. Witte

Illinois Institute of Technology

This work was supported by the Office of Naval Research under contract no. N00014-80-C-0364.

## ABSTRACT

Due to an ever-increasing selection of machines and widely different charging algorithms, the question often arises "on which machine does a program run most efficiently and/or cheaply". The program's behavior, i.e. its resource demands, on the machines under consideration and their impacts on charges have to be explored. Some basic qualities such as accuracy and correctness of the program's results and the reliability of the system also need consideration. Human factors influence decisions as well. The presentation will discuss a procedure and a system which evaluates these factors and their relationship with each other.

## INTRODUCTION

A number of studies have described approaches for machine selection from a computer center management perspective ((2),(6),(10),(11),(18),(19)). Other work has taken the user point of view, but has only examined measurable technical factors such as response time and transmission rates ((13),(14)). Our procedure is unique in that it considers the general "utility" to the user of each service alternative by assessing a broad class of measureable and unmeasureable factors.

Morgan and Campbell (17) discuss service selection factors and the use of a synthetic program model to evaluate the computer systems, but we are unable to use synthetic models because one of our selection factors is the correctness of the application software. In our procedure we run typical production jobs on each different system.

Bell (1) points out that "experimental comparisons in this area are bound to be either crude or very expensive." We have tried to make the comparison sophisticated while keeping the effort and cost as small as possible.

Utility theory offers a wide variety of assessment models (e.g. (5),(7)-(9),(11),(12),(23)) for multifactor decision making. For a textbook treatment of the topic see (11). Because a direct, additive assessment model is generally reliable and easy to use, we will use it to estimate the "utility" of each alternative (9).

As an example, the decision making procedure is used during a study comparing minicomputers and mainframes in the solution of structural analysis problems. We are interested in the relative cost/utility of the two types of machines for processing a representative set of structural analysis problems.

The user's goal when comparing service alternatives is to determine which service is "better" than the others. This is done by comparing cost and utility of each of the services. The "best" alternative will maximize utility while minimizing cost. Our methodology consists of the following steps:

1) design of the evaluation experiment
2) benchmark
3) cost evaluation
4) evaluation of other performance variables
5) weighting and scoring
6) analysis

In order to measure "utility", the user must determine what specific performance variables are most important to him. After the performance variables are selected, benchmark jobs are prepared and run to measure the cost/utility relationship. Overall utility is measured by weighting each performance variable to reflect its relative importance and summing each weighted variable into a utility index. Once the utility index is determined for each alternative, the cost/utility relationships of all alternatives are compared, tradeoffs are made, and the "better" alternative is chosen.

The detailed analysis is centered around benchmark jobs run at IIT on the UNIVAC 1100/81 mainframe system and on the PRIME 400 minicomputer system. We are not only comparing the two "machines" in a limited technical sense, but we are also comparing two computer service environments. The two computer services used in the analysis are typical in many respects to what users will find in local minicomputer or mainframe services because of their workload, charging algorithms, and support staff.

## DESIGN OF THE EVALUATION EXPERIMENT

As part of the design, performance variables must be chosen. The "utility" of a service to a user is a function of those performance factors which affect the value and usefulness to the user. Factors chosen should evaluate the complete computer service and not just the machine itself. The "utility" may include measureable factors such as turnaround time, correctness, and availability, as well as nonmeasureable, subjective factors such as support and ease of use. A list of possible performance variables is in Appendix A. For an older checklist of relevant aspects of a computer see (16). These 84 criteria help in the value assessment when buying a computer, they are less appropriate from a user's perspective. In order to keep the evaluation manageable, the user should keep the number of factors as small as possible, avoiding irrelevant and marginally important variables. Also, care should be taken to keep the variables as independent as possible, i.e. variables should be measuring distinct and separate performance areas.

We have chosen the following performance variables as being most important in determining our "utility" of the computer service alternatives:

1) correctness of results
2) turnaround time
3) system availability or "uptime"
4) support - technical help, documentation

Correctness is important because of its great impact upon usefulness of the results. Turnaround time is important because we normally are waiting for results from the job being run and cannot proceed with further work until the current job is returned. Availability is also important because of the importance of the work and the value of time. Support also affects "utility" because frequently the need for help arises on problems that cannot readily be solved by the user.

Part of the design phase involves planning the benchmarks. The benchmark data are later used for job cost predictions on the machines using interpolation and extrapolation of experimental results. Benchmark data are used to predict job costs on other mainframes and minicomputers. These various uses for the benchmark data affects the kind of data we need and how we collect it.

A set of experimental variables is required to depict a typical structural analysis user and his environment. In addition to varying the machines (mini, mainframe), a representative workload needs to be defined. For an evaluation of test workload generation see (15). We use different finite element programs (SAPIV, SPAR, TWODEL), different problem types (beam, plate, and cylinder), different analysis methods (displacement and stresses, buckling loads and vibration modes, large deformations, transient dynamic analysis), and different problem sizes (varying numbers of elements). The different programs are used so that several typical finite element systems can be compared. The different types and analyses are chosen to represent various problems typically solved in structural analysis applications. The different problem sizes are run to allow us to interpolate and extrapolate results through a broad range of typical problem sizes. On the UNIVAC, one job is run for each combination of control variables. On the PRIME, for reasons discussed later, several jobs are run for each combination.

Only a small part of the benchmark results are presented here to illustrate our procedure. For complete benchmark results refer to (22). Users of structural analysis software on minicomputers have conducted several (more limited) studies that point to the economic advantages of using such machines (e.g.,(3),(20)).

## BENCHMARK

Measurement data collected during the benchmark are used for job cost prediction and performance comparison of the two machines. The following statistics are collected for each run:

- CPU time (measured in seconds)
- IO time (measured in seconds)
- memory used (measured in K bytes)
- turnaround time (measured in seconds)
- # of disk IO operations
- job cost (measured in $)

A comprehensive program performance measurement system was developed for the data collection and analysis. As part of this system, a routine called INST will automatically instrument the FORTRAN source programs by inserting CALL statements at the entry and exit points of each module in the system. The CALL statements invoke a routine called REFSTR which will collect performance data. See Figure 1 for an instrumented subroutine.

```
      SUBROUTINE ELT3A4 (A,B)
      CALL REFSTR('ELT3A4','ENTRY')
      .
      .   body of subroutine
      .
      IF(ISTOP.EQ.0) GOTO 99999
      .
      .
99999 CALL REFSTR('ELT3A4','EXIT')
      RETURN
      END
```

Figure 1 - INST sample output

The REFSTR data collection routine logs entries and exits from modules in the system, captures the CPU and IO times at each of these points, and writes this program trace information out to a permanent file in very large blocks. A sample of the program trace data from REFSTR can be seen in Figure 2.

```
ROUTINE            CPU      IO
$MAIN$ ENTRY  11.1694 9.1964
FACMGT ENTRY  11.1704 9.1964
FACMGT EXIT   13.0202 9.1964
LODCMN ENTRY  13.0246 9.1992
LODCMN EXIT   13.0252 9.1992
INPUTJ ENTRY  13.0522 9.2172
INPUTJ EXIT   13.3984 9.4172
ELTYPE ENTRY  13.4028 9.4352
PLANE  ENTRY  13.4034 9.4352
ELT3A4 ENTRY  13.4098 9.4454
PLNAX  ENTRY  13.4104 9.4454
```

Figure 2 - REFSTR sample output

Calibration studies have been performed on REFSTR to determine the CPU and IO overhead which is generated by the data collection itself. This instrumentation overhead is removed during analysis by the ANALYZ program. ANALYZ processes the raw data from REFSTR, removes instrumentation overhead, and prints summary performance results (see Figure 3).

```
MODULE    CPU     %CPU      IO     %IO    #CALLS  #RETURNS
$MAIN$  0.065     .08     .007     .04       1        1
FACMGT  1.849    2.38     .0       .0        1        1
ELAW    0.139     .18     .128     .48     546      546
ADDSTF 21.396   27.51    9.023   34.09       1        1
.
.
.
SESOL  31.583   40.61   10.213   38.59       1        1
STRSC    .944    1.21    1.255    4.74    1092     1092

TOTAL  77.769           26.467

CPU  = amount of CPU time spent in module
%CPU = fraction of total CPU time spent in module
IO   = amount of IO time spent in module
%IO  = fraction of total IO time spent in module
#CALLS = number of entries into module
#RETURNS = number of exits from module
```

Figure 3 - ANALYZ resource consumption output

ANALYZ also produces a dynamic call matrix to be used for further analysis or program restructuring for virtual memory systems (see Figure 4).

```
                       CALL MATRIX
       TO: 1    2    3    4    5    6    7    8    9   10
 FROM:
 1 $MAIN$ 0    1    1    1    1    0    0    0    0    0
 2 FACMGT 0    0    0    0    0    0    0    0    0    0
 3 LODCMN 0    0    0    0    0    0    0    0    0    0
 4 INPUTJ 0    0    0    0    0    0    0    0    0    0
 5 ELTYPE 0    0    0    0    0    2    0    0    0    0
 6 PLANE  0    0    0    0    0    0    1    0    0    0
 7 ELT3A4 0    0    0    0    0    0    0    1    0    0
 8 PLNAX  0    0    0    0    0    0    0    0  546    0
 9 ELAW   0    0    0    0    0    0    0    0    0  546
10 POSINV 0    0    0    0    0    0    0    0    0    0
   .
   .
   .
```

Figure 4 - ANALYZ call matrix output

A routine called LSP was developed to fit polynomial curves to the resource consumption and cost data using a least squares technique. LSP also can plot the data points and fitted curves (see Figure 5).

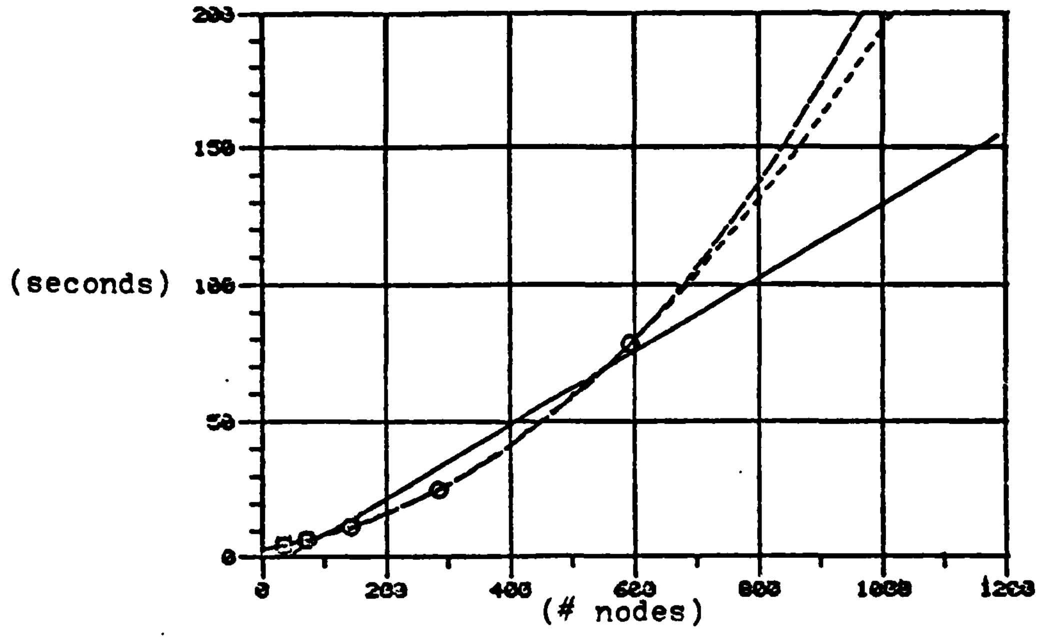

Figure 5 - CPU time vs problem size (PLATE)

The benchmark performance data is collected at the job level for all jobs, at the program level for all SPAR jobs, and at the subroutine level for a few selected SAPIV jobs. Since SPAR is a modular sequence of rather independent tasks, it was not necessary to collect measurement data at the subroutine level. SAPIV, on the other hand, is not sequentially structured, but is a conglomerate of subroutines with frequent interaction. This necessitates more detailed performance data collection from which SAPIV's resource consumption models are derived (see Figure 3). For the purposes of job cost prediction alone, the job level resource consumption data turned out to be sufficient in either case.

The resource consumption statistics are collected differently on each machine. The UNIVAC operating system easily provides the necessary job level data through job accounting printouts and system commands. On the PRIME, a FORTRAN program collects program level data. It uses system subroutines which return CPU and IO times. To measure performance at the subroutine level, FORTRAN subroutines which access standard system subroutines collect the necessary data. Instrumentation overhead is measured and removed from the data, if it is significant. The IO counts are provided by SPAR itself through internal instrumentation already available.

Job costs are computed based upon resource consumption and are not taken directly from the job accounting output. This is done so that we can exclude unit record (card reader, line printer) IO costs, investigate and compare differential (priority, time-of-day) rate schedules, and thoroughly understand and validate the charging algorithm.

## COST EVALUATION

To assess the cost-effectiveness of a machine, it is necessary to thoroughly understand the charging algorithm used. The data collected from the benchmark jobs tells us how much of each resource is used, but it is only after we apply the charging algorithm that we find which resources most directly affect the cost of the job. Differences in machines, operating systems, workloads, management strategies, etc influence how the use of various resources is charged and explain the wide variation of charging algorithms. Users will find that some resources are not charged for directly, such as IO on the UNIVAC and memory on the PRIME. This may, although not necessarily so, indicate that use of that resource is relatively cheap or even free. In many cases a "free" resource is not really free because it is being charged for indirectly through the associated use of other "not-so-free" resources. By knowing the resource consumption characteristics of one's typical workload, and by understanding what resources are relatively expensive and which are not, the user should be able to determine which machine provides the "better", lower overall cost.

By looking at the following two charging algorithms, we can easily see how CPU, IO, and MEMORY resources are charged. The following charging algorithms are from (4):

PRIME cost ($) = .02 * (CPU sec) + .02 * (IO sec)
+ 1.00 * (connect hours)

UNIVAC cost ($) = .18 * (CPU sec)
+ .0011 * (#memory blocks) * (CPU + IO sec)

Because the PRIME charges directly for CPU and IO resources only, charges on the PRIME are somewhat insensitive to large memory requirements (IO for paging offsets this in heavily loaded environments.) Because the UNIVAC charges directly for CPU and MEMORY only, charges on the UNIVAC are fairly insensitive to large IO resource consumption provided that only a small amount of memory is used. Therefore, the machines are cost-sensitive to different kinds of resources.

For each job, the resource consumption data are combined with the charging algorithm for the machine and resulting costs are determined. The relationship between cost and problem size is important because this relationship will predict jobs costs over a wide range of problem sizes. Through least squares fitting of a polynomial curve, we are able to determine and plot the functional relationship between cost and problem size for each combination of machine, program, and problem type. Curve fitting and curve plotting is done for each combination using the LSP curve fitting routine in our system. When cost vs problem size fitting is done most of the beam cost functions turned out to be highly linear, while the plate and cylinder problems exhibited slight quadratic and cubic tendencies. Figure 6 is an example of the curve fit for SPAR beam problems on the PRIME. For a particular problem, benchmark results and fitted curves can be combined for comparison of the same program on different machines (Figure 7) or different programs on the same machine (Figure 8).

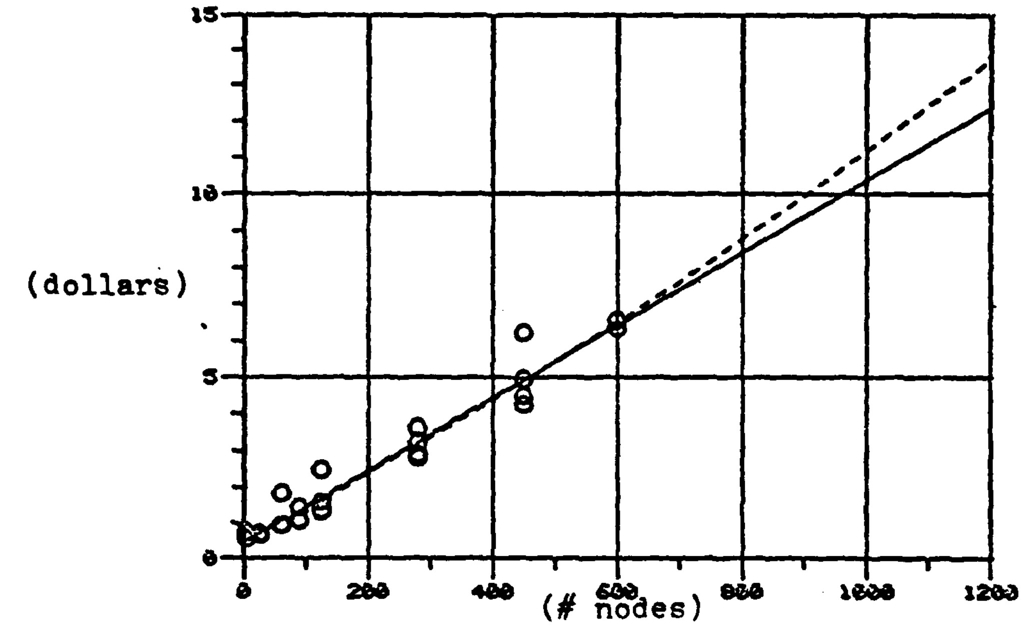

Figure 6 - Cost vs problem size (BEAM)

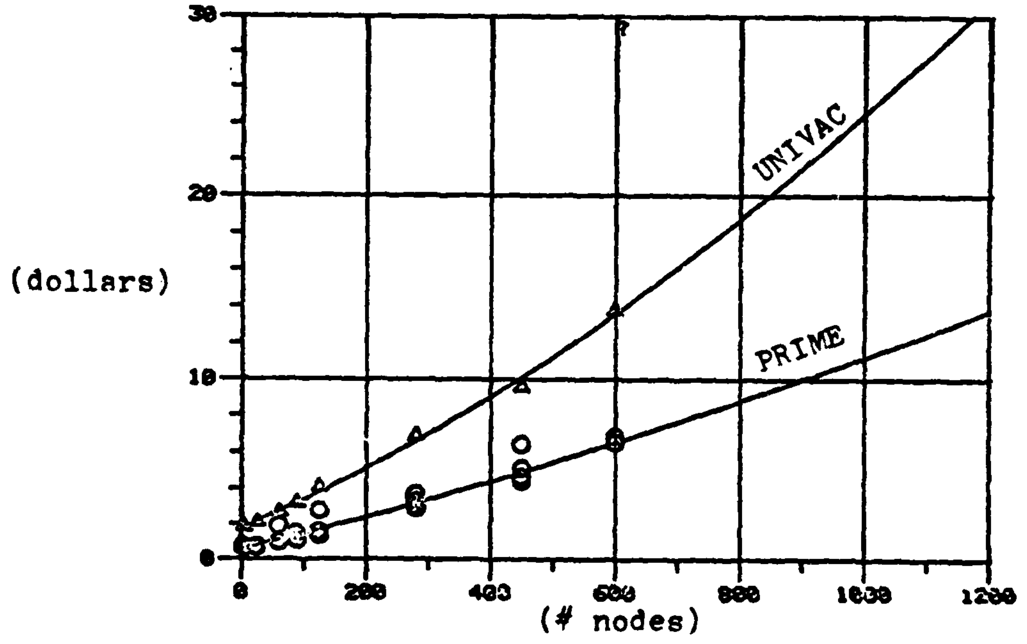

Figure 7 - Cost vs problem size (BEAM)

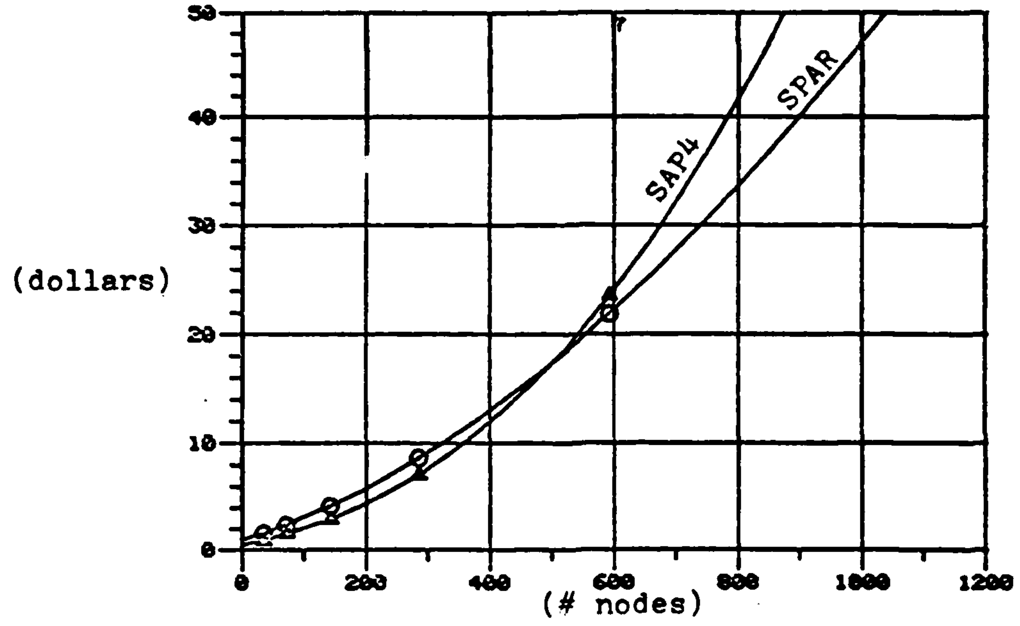

Figure 8 - Cost vs problem size (PLATE)

On the UNIVAC, resource consumption and job cost are reproducible because OS1100 does not charge the user for memory management overhead due to swapping.

On the PRIME, resource consumption and job cost varies widely for different runs of the same job. In situations where memory contention is high, IO time increases greatly due to paging. In other cases, when processor contention is high, CPU times increase significantly. In extreme cases the resource consumption (and job cost) varies by a factor of 3:1. This is due to the fact that PRIMOS charges the user for all IO operations including paging and for certain CPU overhead due to processor scheduling. To make matters worse, there is no differentiation in job accounting data to indicate which resources are for actual user work and which are for overhead. At least three jobs are run on the PRIME for each combination of variables. The three jobs are run in low, medium, and high system load environments in order to identify a range of performance characteristics approximating best and worst cases. The curve fitting is done using all data points, thereby averaging the results.

The outcome of the cost analysis indicates that the PRIME costs about half as much as the UNIVAC for small, simple problems such as the beam (see Figure 7.) However, for larger, more complex problems (plate) the PRIME costs as much as 35% more than the UNIVAC when the SPAR package is used (see Figure 9.)

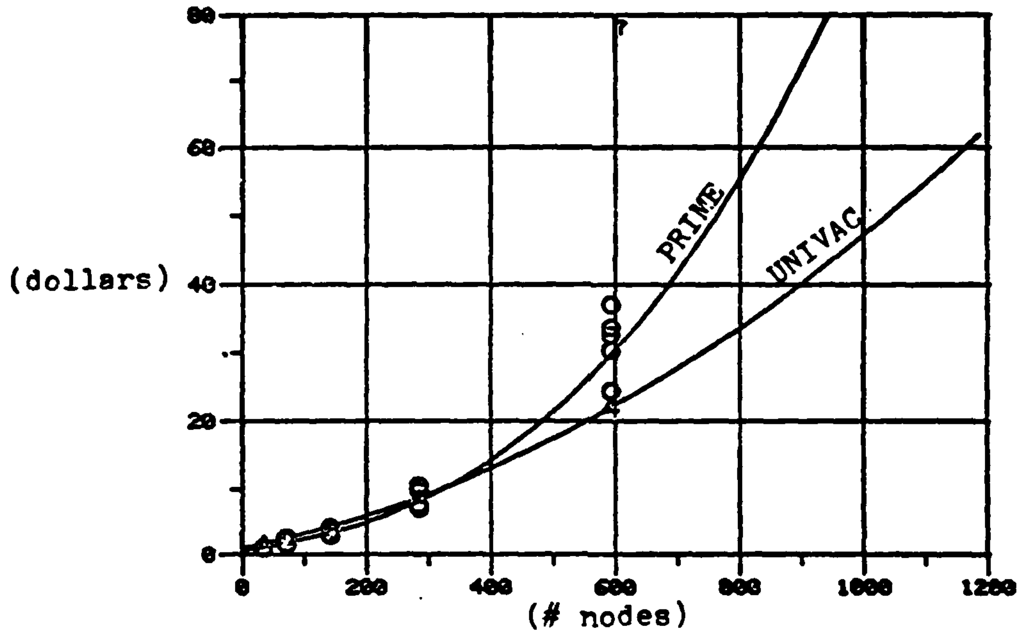

Figure 9 - Cost vs problem size (PLATE)

## OTHER PERFORMANCE VARIABLES

Correctness of results is so important that most people assume it. There are several reasons why assuming correct results may be dangerous. In certain problems which require high precision floating point values to be stored and manipulated, the minicomputer may store and manipulate fewer significant digits than the mainframe because of smaller word sizes and floating point operand sizes. This may produce completely incorrect results or poor results at best. Another potential problem is the correctness of the application program. Packages that have been developed on one machine and subsequently transported to another may have been converted improperly. We found several problem types that we could not run correctly on the PRIME due to those problems. About 30% of the beam runs and 50% of the plate runs were completely correct when run on the PRIME. The most important thing that a user can do to protect himself is to check results carefully and validate them if possible.

Turnaround time is important to most users and is measured as part of the data collection for each job. See Figures 10 and 11. Turnaround times on the UNIVAC vary from about 5 minutes to about 60 minutes for the various jobs. The turnaround times on the PRIME vary from 3 minutes to about 10 hours. Small jobs run quickly on both machines, but large jobs tend to take much longer on the PRIME than on the UNIVAC. As an example, the 594 node SPAR plate problem took between 20 and 70 minutes on the PRIME and about 5 minutes on the UNIVAC.

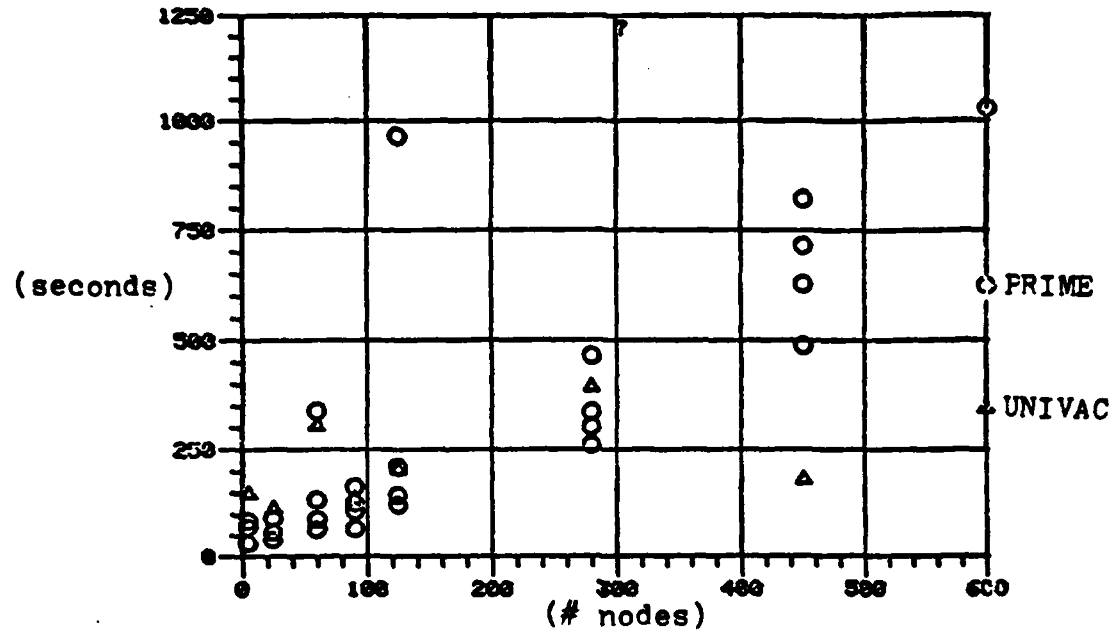

Figure 10 - Wall time vs problem size (BEAM)

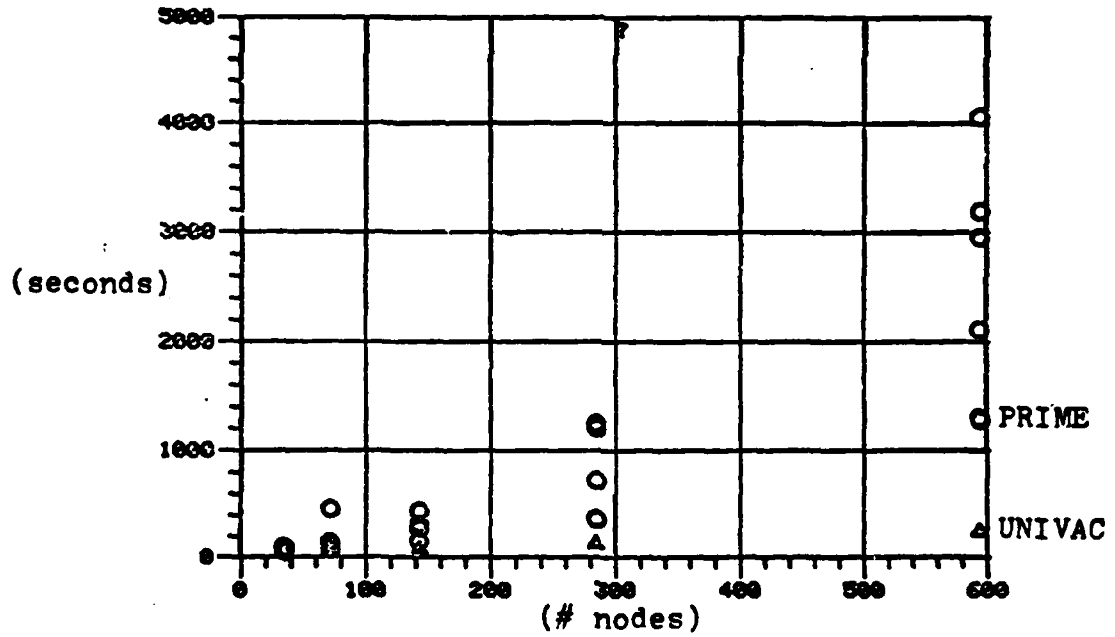

Figure 11 - Wall time vs problem size (PLATE)

Availability or "uptime" is important to us and is determined based upon data provided by the two centers. They are:

| | UNIVAC | PRIME |
|---|---|---|
| UPTIME % | 98.9 | 96.7 |
| MTBF (hours) | 56.79 | 87.98 |
| MTTR (minutes) | 34.7 | 178.2 |

The UNIVAC data is taken from published figures for January 1980 through December 1980. The PRIME data could only be obtained for the months of October through December 1980. The PRIME system is down fewer times than the UNIVAC, but is down for extended periods of time. The overall uptime of the UNIVAC is somewhat better than the PRIME.

Support is a very important performance factor for many users. Support includes availability and quality of technical help and documentation. We found both the UNIVAC and PRIME centers to be somewhat deficient in support. We found the UNIVAC documentation to be fairly good, the access to documentation to be good, but the access to technical help to be poor (due to contract between service bureau and university). We found the PRIME documentation to be weak and unavailable for reference except by purchase. Particularities of the IIT PRIME 400 system are not documented at all in the standard manuals and are only communicated by word of mouth. Although the technical people are fairly competent, they are far overburdened and unable to provide a consistent level of support.

## WEIGHTING AND SCORING

The relative importance of the performance variables is reflected by assigning weights to each variable. The weight values are arbitrary in absolute magnitude but extremely important with respect to relative magnitude when compared to each other. A possible weighting scale is shown in Appendix B. We have assigned the following weights to our four performance variables:

| | |
|---|---|
| correctness of results | 5 |
| turnaround time | 3 |
| system availability | 3 |
| support | 2 |

Each of the raw performance observations should be converted to a performance score for each particular variable. The performance score should reflect the relative value to the user of that particular raw score for the variable. The conversion is done by direct transformation of raw data or by subjective evaluation and grouping of results. Example scoring scales are shown in Appendix C.

The weights and scores are tabulated and evaluated to determine the utility index. We can easily compare the UNIVAC and

PRIME if we choose a particular program, problem type, and size. For a comparison of utility vs cost for the two machines running SPAR beam and plate problems see Figures 12 and 13. These figures show that, although the PRIME may be cheaper in many situations, the utility of the PRIME is significantly lower than that of the UNIVAC. In addition to comparing the relative utilities of alternatives, it is also interesting to compare each utility with the maximum possible utility (in our case 65).

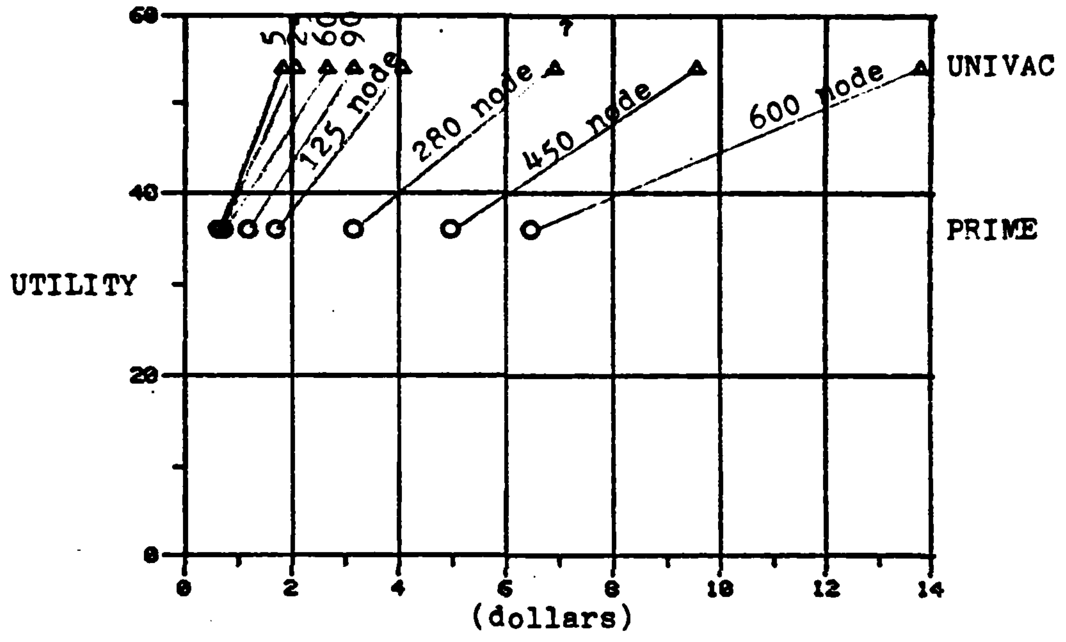

Figure 12 - Utility vs problem size (BEAM)

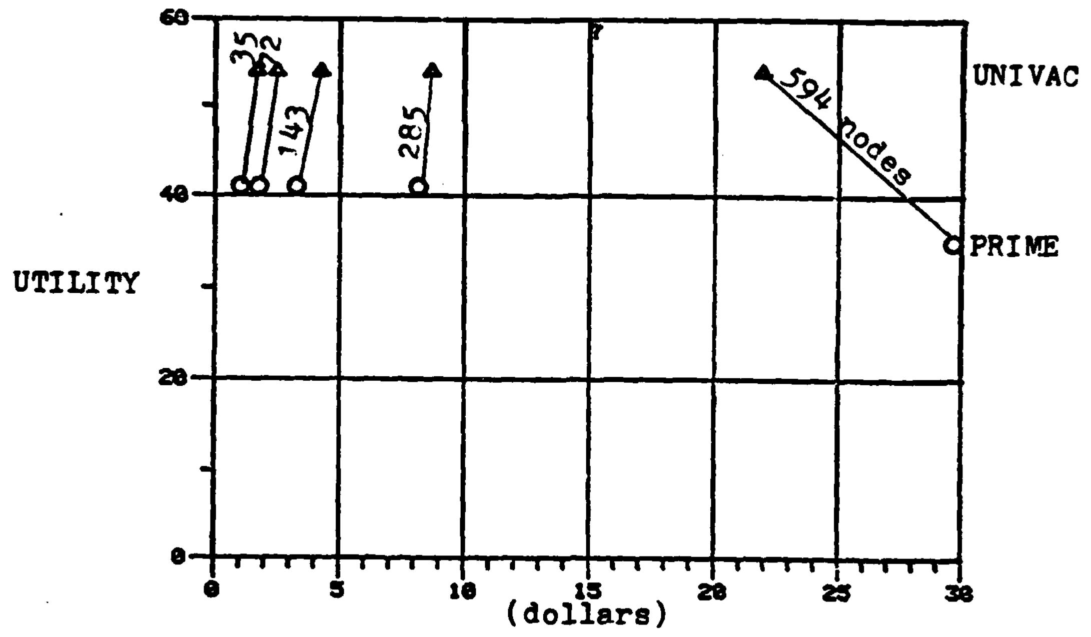

Figure 13 - Utility vs problem size (PLATE)

ANALYSIS

For beam problems the PRIME is about 50% the cost of the UNIVAC. However, due to problems with correctness of the SPAR package on the PRIME, the utility of the PRIME system is much lower than the UNIVAC which makes the UNIVAC the better system to use.

For plate problems, the costs are similar with the UNIVAC being somewhat (35%) cheaper on the larger problems. The UNIVAC is again the best system to use because of the similar cost and much higher utility.

We can see from the examples that the type of problem and software may be an important factor in determining which machine is "better" to use. Users will have to project the frequency with which they use certain software, problem types, and problem sizes to combine the costs and utilities for their particular profile of use. The cost and utility figures should be weighted by the frequency with which they occur and combined into an aggregate cost and utility measurement for the projected workload.

CONCLUSION

We have described a comprehensive approach to evaluate the cost and utility of alternative computer services. The utility is evaluated by using technical and human factors. This approach will provide the user with a much better base of information for making a choice between alternatives than provided by traditional cost vs throughput studies. The strength of this approach is that it considers a broad range of relevant factors, it is simple to understand and use, and it allows the user to compare the cost/utility relationships of computing alternatives so that tradeoffs can be considered.

As in most complex decision making processes, several major aspects of our methodology are necessarily subjective. The selection of performance factors is subjective and could be the subject of considerable discussion, analysis, and review. We have not dealt with various methods for experimentally identifying factors and their interdependence because of the tremendous amount of effort involved.

The selection of relative weights and scoring levels is also subjective. The weights and scores should be carefully chosen by the user so that they reflect the needs and requirements for that user. The weighting and scoring can only be as good as the judgement of the user who is evaluating a given situation.

Using experimental techniques to select factors, weights, and scores goes far beyond the scope of our paper. The combination of expert judgement for decision making parameters with an objective decision making procedure results in a practicable comprehensive evaluation approach.

## APPENDIX A

PERFORMANCE VARIABLES

- availibility
- reliability
- capacity
- compatibility
- documentation
- languages
- operating system features
- response time
- turnaround time
- technical staff
- ease of use
- access (batch, interactive)
- correctness of results
- availability of applications software

## APPENDIX B

PERFORMANCE VARIABLE WEIGHTING

5 - critical
4 - very important
3 - important
2 - somewhat important
1 - slightly important
0 - unimportant

## APPENDIX C

PERFORMANCE VARIABLE SCORING

| Correctness of Results | |
|---|---|
| 100% correct | 5 |
| 80% correct | 4 |
| 60% correct | 3 |
| 40% correct | 2 |
| 20% correct | 1 |

Turnaround Time

| | |
|---|---|
| 15 minutes | 5 |
| 30 minutes | 4 |
| 45 minutes | 3 |
| 60 minutes | 2 |
| ¨60 minutes | 1 |

System Availability

| | |
|---|---|
| 99% - 100% | 5 |
| 98% - 99% | 4 |
| 95% - 97% | 3 |
| 91% - 94% | 1 |
| less than 91% | 0 |

Support

| | |
|---|---|
| excellent | 5 |
| good | 4 |
| fair | 3 |
| poor | 1 |
| none | 0 |

## BIBLIOGRAPHY

(1) Bell, T. E., "Computer performance variability," AFIPS Conference Proceedings 43 (NCC), 1974, 761-766.

(2) Brocato, L. J., "Getting the best computer system for your money," Computer Decisions 3, 9(September 1971), 32-39.

(3) Conaway, J. H., "The Economics of Structural Analysis of Superminis", Proceedings 7th ASCE Conference of Electronic Computation, 1979, 374-385.

(4) Dix, Rollin C., "Academic Computing at IIT," Academic Computing Office, Illinois Institute of Technology, Chicago, IL, September 1980.

(5) Efstathiou, J., and Rajkovic, V., "Multiattribute Decisionmaking Using a Fuzzy Heuristic Approach," IEEE Transactions on Systems, Man, and Cybernetics 9, 6(1979), 326-333.

(6) Ferrari, D., Computer Systems Performance Evaluation, Prentice-Hall, Englewood Cliffs, NJ, 1978.

(7) Grochow, J. M., "A Utility Theoretic Approach to Evaluation of a Time-Sharing System," Statistical Computer Performance Evaluation, W. Freiberger (Ed.), Academic Press, New York, 1972.

(8) Grochow, J. M., "Utility functions for time-sharing system performance evaluation," Computer 5, 5(September/October 1972), 16-19.

(9) Johnson, Edgar M., and Huber, George P., "The Technology of Utility Assessment," IEEE Transactions on Systems, Man, and Cybernetics SMC-7, 5(May 1977), 311-323.

(10) Joslin, E. O., Computer Selection, Addison-Wesley, Reading, MA, 1968.

(11) Kleijnen, J. P. C., Computers and Profits - Quantifying Financial Benefits of Information, Addison-Wesley, Reading, MA, 1980.

(12) Kleijnen, J. P. C., "Scoring Methods, Multiple Criteria, and Utility Analysis," Performance Evaluation Review 7, 4(1980).

(13) Mamrak, S. A., and Amer, P. D., "Comparing interactive computer services - theoretical, technical, and economic feasibility," Proceedings of the National Computer Conference - 1979, 781-787.

(14) Mamrak, S. A., and DeRuyter, P. A., "Statistical Methods for the Comparison of Computer Services," Computer 10, 11(November 1977), 32-39.

(15) Mamrak, S. A., and Abrams, M. D., "A Taxonomy for Valid Test Workload Generation," Computer 12, 12(December 1979), 60-65.

(16) Miller, W. G.,"Selection Criteria for Computer System Adoption," Educational Technology 9, 10(1969), 71-75.

(17) Morgan, D. E., and Campbell, J. A., "An answer to a user's plea?", Proceedings 1st ACM-SIGME Symposium on Measurement and Evaluation, February 1973, 112-120.

(18) Randal, J. M., and Badger, George F., "Using Quantitative Criteria for Computer Selection," Computing Services Office, University of Illinois at Urbana-Champaign, Urbana, IL, 1976.

(19) Sharpe, W. F., The Economics of Computers, Columbia University Press, New York, 1969.

(20) Storaasli, O. O., and Foster, E. P., "Cost-Effective Use of Minicomputers to Solve Structural Problems," 1978, 164-169.

(21) Timmreck, E. M., "Computer selection methodology," Computing Surveys 5, 4(1973), 199-222.

(22) Von Mayrhauser, A. K., and Haftka, R. H., "The Cost Effectiveness of Minicomputers vs. Mainframes in the Solution of Structural Analysis Problems," TR81-01, Department of Computer Science, Illinois Institute of Technology, Chicago, IL, March 1981.

(23) Zadeh, L. A., "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," IEEE Transactions on Systems, Man, and Cybernetics 3, 1(1973), 28-44.